Vol. 6 No. 1, 2025, pp. 92-102

DOI: 10.23960/pepadun.v6i1.265

Jurnal Pepadun

Determining the Shortest Tour Location of Tourist Attractions in Bandar Lampung Using Cheapest Insertion Heuristic (CIH) and Modified Sollin Algorithm

[1]Abdi Restu Dinata, *[2]Wamiliana, [3]Muslim Ansori, [4]Fitriani and [5]Notiragayu

[1,2,3,4]Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Lampung,
Jl. Prof. Soemantri Brojonegoro No.1, Kota Bandar Lampung
e-mail: [1]abdirstu02@gmail.com, *[2]**wamiliana.1963@fmipa.unila.ac.id**, [3]muslim.ansori@fmipa.unila.ac.id,
[4]fitriani.1984@fmipa.unila.ac.id, [5]notiragayu@fmipa.unila.ac.id

Abstract - *The travel and tourism industry play important role in economics. Like various urban areas on the island of Sumatra which is famous for its tourist destinations, the city of Bandar Lampung is one of the tourist destinations for urban communities on the island of Sumatra because it has many cultural tourist attractions that tourists can visit. With so many choices of tourist destinations, tourists will definitely think about considering the time and costs as efficiently as possible to visit the available tourist attractions. Therefore, it is necessary to take the shortest tour so that it can save time and costs. This problem is known as the Traveling Salesman Problem (TSP). In this study the Cheapest Insertion Heuristic and Modified Sollin Algorithm are used to solve the problem. The results obtained show that the solution using the modified Sollin Algorithm is better than the Cheapest Insertion Heuristic*.

Keywords: *Travelling Salesman Problem; Cheapest Insertion Heuristics; Modified Sollin Algorithm; Tourist Attractions.*

1. INTRODUCTION

The tourism travel industry is one of the important industries that influence the economic growth. Like various urban areas on the island of Sumatra and famous for its tourist destinations, Bandar Lampung City is one of the tourist destinations for urban communities on the island of Sumatra because it has many tourist and cultural attractions that can be visited by tourists. With so many choices of tourist destinations, tourists will definitely think about considering the time and cost as efficiently as possible to visit the available tourist attractions. If the tourist wants to visit every tourist spot and then return to the place where he started his journey with the minimum distance or cost, this problem is known as the Traveling Salesman Problem (TSP). Problem solving efforts on tour determination are often represented as the TSP, where the TSP is a classic combinatorial optimization problem in computer science and mathematics to find a tour that visits each location once and returns to the starting point [1].

The TSP is widely used and regarded as one of the traditional network design challenges. Willian Rowan Hamilton, an Irish mathematician, developed the mathematical formulation of that problem. In 1856, he developed the mathematical game known as the Icosian game. The object of that game is to find a Hamiltonian cycle, a cycle on a dodecahedron that passes through each vertex using the dodecahedron's edges. TSP gained traction in American and European scientific circles in the 1950s and 1960s after efforts to address it were considered for prizes by the Santa Monica-based RAND Corporation [2]. Usually, graph *G (V,E)* are used to represent TSP problem, where the set of vertices $V = \{v_1, v_2, \dots, v_n\}$ represents the cities, while the edges $E = \{e_{ij} | i, j \in V\}$ represent the road, and associated with every edge there is $c_{ij} \geq 0$ which represents distance, cost, time, etc.

The TSP is highly investigated, and due to its NP hard complexity, heuristics are more preferable to be investigated instead of exact methods. Some of the methods that already used to solve the TSP. Kurniawan et al [3] used Particle Swarm Optimization (PSO) dan Brute Force to solve the TSP and implemented using the data with up to 30 vertices. Violina [4] used Brute Fore method and Branch and Bound to solve the TSP. The solution using Brute and Force always optimal but needs quite along time because it calculates all possibilities. In contrast, Branch and Bound obtains the best solution more quickly because it does not compute all possibility. To solve the TSP, Wilson et al [5] used Brute and Force with Graphic Processing Unit. Amelia et

Vol. 6 No. 1, 2025, pp. 92-102 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.265

al [6] compared Brute Force, Cheapest-Insertion, and Nearest-Neighbor Heuristics for Determining the Shortest Tour for Visiting Malls in Bandar Lampung and show the optimal solution was gained by Brute and Force Method, but inefficient in terms of time processing. The Nearest Neighbour Heuristic also used by Hougardy and Wilde [7] for the metric TSP, and Winda et al [8] used Nearest Neighbour Heuristic to find the best route for product distribution.

The Cheapest Insertion Heuristics is used by Aswin [9] to find the best tour for traditional markets in Bandar Lampung city. The Cheapest Insertion Heuristic is also used by Hignasari and Mahira [10], Meliantri et al [11], and Utomo et al [12] for finding product distribution. Kusrini and Istiyanto [13] illustrated the method using simple example of 5 cities. The Christofides Algorithm is used by Aswin et al [9] to find the traditional markets tour, and by Tjoea and Halim[14] to find the ship voyage route evaluation. In this research, a comparison will be made between CIH and the modification of Sollin's algorithm to determine the TSP solution for tourist objects in Bandar Lampung City.

2. RESEARCH METHODOLOGY

2.1 The Cheapest Insertion Heuristics and The Modification of Sollin's Algorithm

A technique called the Cheapest Insertion Heuristic (CIH) iteratively constructs a tour by adding the cheapest (least expensive) node to the existing partial tour in order to approximate a solution to the Traveling Salesman Problem (TSP). In order to discover the cheapest way to integrate a remaining node into the tour, it first starts with a small tour. We do make a small modification in CIH algorithm, where in the first step we make tour of three vertices instead of two vertices. The procedure of CIH algorithm is given as follow:

```
Given a graph G(V,E) with n vertices and m edges.
      Step 1    : Make a small tour that only consists of three vertices.
      Step 2    : Search for candidate edge to be merged with the small tour in order
                  to make a new subtour.
      Step 3    : Calculate all selected candidate edges using the formula:
                  Total current weight = Total weight - weight of the discarded edge
                  + weight of the added edge + weight of the edge that merge the added
                  vertex with the vertex in the previous subtour.
      Step 4    : Select the least weighted value from the calculated candidate edges.
      Step 5    : Insert the selected edge in Step 4 to make the new subtour.
      Step 6    : If the number of vertices in the subtour in Step 5 = n, then stop.
                  Otherwise, back to Step 2.
```

2.2 The Modification of Sollin's Algorithm

Sollin's algorithm is commonly used to determine the Minimum Spanning Tree (MST) of a connected graph [15]. However, by making modifications, Sollin's Algorithm can be used to solve the TSP. The following steps are given to modify Sollin's Algorithm to solve the TSP.

```
      Step 1    : Connect each vertex in the graph with the smallest edge weight that
                  incident to it.
      Step 2    : For vertex with degree more than two, reduce the degree by removing
                  one of the edges to form an isolated vertex, and then connect the
                  isolated vertex with a leaf (vertex of degree one).
      Step 3    : Connect all leaves so that all leaves become vertices of degree two
                  (while also checking the smallest edge when doing the connection
                  process, and also avoiding to form cycle). Do this process until the
                  degree of every vertex is two.
```

2.3 The Data

Tabel 1 shows the time (in minute) needed to go from one location to other locations, and the photos of the 24 tourist sites are given in Table 2.

Vol. 6 No. 1, 2025, pp. 92-102
Jurnal Pepadun

DOI: 10.23960/pepadun.v6i1.265

Table 1. The time needed (in minute) to go from place v_i to $v_j, i, j = 1,2,3, ...,24$.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}
v_1	0	22	15	16	21	36	20	39	37	28	18	17	14	16	24	16	25	18	33	32	22	29	29	20
v_2	22	0	18	16	17	22	6	24	39	10	15	33	31	23	5	23	4	13	12	11	1	7	15	17
v_3	15	18	0	15	18	34	13	36	43	27	16	26	26	24	21	18	22	15	29	29	19	26	27	17
v_4	16	16	15	0	10	26	15	29	38	19	8	25	21	19	18	13	20	11	25	24	17	23	20	10
v_5	21	17	18	10	0	24	17	27	35	17	6	29	24	16	21	14	21	9	22	21	19	22	17	5
v_6	36	22	34	26	24	0	26	7	33	12	18	33	30	20	26	22	24	25	18	17	21	17	11	19
v_7	20	6	13	15	17	26	0	30	42	16	13	30	26	25	7	20	10	11	18	17	7	13	22	15
v_8	39	24	36	29	27	7	30	0	37	16	23	37	33	23	29	26	26	29	22	21	24	21	15	23
v_9	37	39	43	38	35	33	42	37	0	31	30	25	26	23	44	30	41	37	35	34	38	36	25	31
v_{10}	28	10	27	19	17	12	16	16	31	0	13	28	25	14	15	16	13	20	7	6	10	7	7	13
v_{11}	18	15	16	8	6	18	13	23	30	13	0	23	19	12	17	9	19	9	17	16	15	17	12	2
v_{12}	17	33	26	25	29	33	30	37	25	28	23	0	4	17	33	18	34	28	29	29	30	30	24	23
v_{13}	14	31	26	21	24	30	26	33	26	25	19	4	0	13	32	15	32	27	29	28	29	29	22	19
v_{14}	16	23	24	19	16	20	25	23	23	14	12	17	13	0	26	8	23	18	18	17	21	18	12	11
v_{15}	24	5	21	18	21	26	7	29	44	15	17	33	32	26	0	23	9	14	17	16	5	12	21	19
v_{16}	16	23	18	13	14	22	20	26	30	16	9	18	15	8	23	0	24	17	21	20	21	21	14	10
v_{17}	25	4	22	20	21	24	10	26	41	13	19	34	32	23	9	24	0	13	14	13	4	10	18	20
v_{18}	18	13	15	11	9	25	11	29	37	20	9	28	27	18	14	17	13	0	24	23	15	19	19	8
v_{19}	33	12	29	25	22	18	18	22	35	7	17	29	29	18	17	21	14	24	0	2	11	9	12	18
v_{20}	32	11	29	24	21	17	17	21	34	6	16	29	28	17	16	20	13	23	2	0	10	8	12	17
v_{21}	22	1	19	17	19	21	7	24	38	10	15	30	29	21	5	21	4	15	11	10	0	7	16	17
v_{22}	29	7	26	23	22	17	13	21	36	7	17	30	29	18	12	21	10	19	9	8	7	0	13	18
v_{23}	29	15	27	20	17	11	22	15	25	7	12	24	22	12	21	14	18	19	12	12	16	13	0	12
v_{24}	20	17	17	10	5	19	15	23	31	13	2	23	19	11	19	10	20	8	18	17	17	18	12	0

Description:

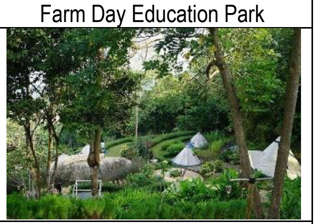
v_1 = Museum Lampung
v_2 = Taman Betung
v_3 = Lembah BKP
v_4 = Bukit Sakura
v_5 = Puncak Mas
v_6 = Pintu Langit
v_7 = Lengkung Langit
v_8 = Pulau Permata
v_9 = Pantai Tiska
v_{10} = Wira Garden
v_{11} = Lembah Hijau
v_{12} = Transmart
v_{13} = Lampung Walk
v_{14} = Lampung Elephant Park
v_{15} = Tebing Vietnam
v_{16} = Teropong Kota Bukit Sindy
v_{17} = Lembah Durian Farm
v_{18} = Camp 91 Kedaung Outbound
v_{19} = Puncak Nirwana
v_{20} = Farm Day Education Park
v_{21} = Taman Kupu-kupu Gita Persada
v_{22} = Air Terjun Batu Putu
v_{23} = Waterpark Citra Garden
v_{24} = Alam Wawai

Table 2. The photos of the 24 tourist sites.

			
Museum Lampung	Taman Betung	Lembah BKP	Bukit Sakura
			
Puncak Mas	Pintu Langit	Lengkung Langit	Pulau Permata

Vol. 6 No. 1, 2025, pp. 92-102

DOI: 10.23960/pepadun.v6i1.265

			
Pantai Tiska	Wira Garden	Lembah Hijau	Transmart Lampung
			
Lampung Walk	Lampung Elephant Park	Tebing Vietnam	Teropong Kota Bukit Sindy
			
Lembah Durian Farm Stable	Camp 91 Kedaung Outbound	Puncak Nirwana	Farm Day Education Park
			
Taman Kupu-kupu Gita Persada	Air Terjun Batu Putu	Waterpark Citra Garden	Alam Wawai

3. RESULTS AND DISCUSSION

3.1. Solving with Cheapest Insertion Heuristics

The manual process of determining the solution for the TSP of 24 tourist location in Bandar Lampung city are simplified on Table 3. From Table 3 we obtain the solution which is 248 minutes (not include the time staying in the locations to see the view or other purposes) needed to make tour of 24 tourist locations in Bandar Lampung City. The results obtained using Cheapest Insertion Heuristic (CIH) are presented in Figure 1.

Figure 1. The solution obtained manually using CIH.

Vol. 6 No. 1, 2025, pp. 92-102

DOI: 10.23960/pepadun.v6i1.265

Table 3. The solution obtained using CIH manually.

No	Subtour	Total time	$\|E\| = n$?	Edges to consider	Total time for new tour	Description
	$v_1 - v_{13} - v_{12} - v_1$	35	No	$d(v_1, v_{13}) = 14$ $d(v_{13}, v_{12}) = 4$ $d(v_{12}, v_1) = 17$		
1	$v_1 - v_{13} - v_{12} - v_1$	35	No	$d(v_1, v_2) = 22$	$35 - d(v_1, v_{12}) + d(v_1, v_2) + d(v_{12}, v_2) = 73$	Because the addition of edges $e_{1,2}$ and $e_{12,2}$ has the minimal total time then choose $e_{1,2}$ as the edge that to be merged in subtour.
				$d(v_{13}, v_2) = 31$	$35 - d(v_{13}, v_1) + d(v_{13}, v_2) + d(v_1 v_2) = 74$	
				$d(v_{12}, v_2) = 33$	$35 - d(v_{12}, v_1) + d(v_{12}, v_2) + d(v_1, v_2) = 73$	
						New subtour: $v_1 - v_{13} - v_{12} - v_2 - v_1$
2	$v_1 - v_{13} - v_{12} - v_2 - v_1$	73	No	$d(v_1, v_3) = 15$	$73 - d(v_1, v_2) + d(v_1, v_3) + d(v_2, v_3) = 84$	Because the addition of edges $e_{1,3}, e_{12,3}$ dan $e_{2,3}$ has the minimal total time then we choose $e_{1,3}$ as the edge that to be merged in subtour.
				$d(v_{13}, v_3) = 26$	$73 - d(v_{13}, v_1) + d(v_{13}, v_3) + d(v_1, v_3) = 100$	
				$d(v_{12}, v_3) = 26$	$73 - d(v_{12}, v_2) + d(v_{12}, v_3) + d(v_2, v_3) = 84$	
				$d(v_2, v_3) = 18$	$73 - d(v_2, v_{12}) + d(v_2, v_3) + d(v_{12}, v_3) = 84$	New subtour : $v_1 - v_{13} - v_{12} - v_2 - v_3 - v_1$
					....	
22	$v_1 - v_{13} - v_{12} - v_9 - v_{14} - v_{16} - v_{23} - v_6 - v_8 - v_{10} - v_{22} - v_{19} - v_{20} - v_{24} - v_{11} - v_5 - v_4 - v_{18} - v_{15} - v_{21} - v_2 - v_{17} - v_7 - v_3 - v_1$	248	Yes			

*Note that $d(v_i, v_j) = c_{i,j}$.

Since the solution consists intersection path, then the solution obtain are suboptimal and we refine the solution by removing intersection edges and adding other edges to obtain a new tour with better solution. Figure 2 shows the removed edges (in red colour), and the added edges (in green colour). The addition and deletion of edges on the tour is done as follows:

i. Remove $e_{7,17}$ and add $e_{7,15}$. The additional/reduction of time after remove/add process = $-d(v_7, v_{17}) + d(v_7, v_{15}) = -10 + 7 = -3$
ii. Remove $e_{18,15}$ and add $e_{18,17}$. The additional/reduction of time after remove/add process = $-d(v_{18}, v_{15}) + d(v_{18}, v_{17}) = -14 + 13 = -1$
iii. Remove $e_{24,20}$ and add $e_{24,22}$. The additional/reduction of time after remove/add process = $-d(v_{24}, v_{20}) + d(v_{24}, v_{22}) = -17 + 18 = 1$
iv. Remove $e_{22,19}$ and add $e_{22,20}$. The additional/reduction of time after remove/add process = $-d(v_{22}, v_{19}) + d(v_{22}, v_{20}) = -9 + 8 = -1$

Vol. 6 No. 1, 2025, pp. 92-102

DOI: 10.23960/pepadun.v6i1.265

v. Remove $e_{10,22}$ and add $e_{10,19}$. The additional/reduction of time after remove/add process $= -d(v_{10}, v_{22}) + d(v_{10}, v_{19}) = -7 + 7 = 0$
vi. Remove $e_{10,8}$ and add $e_{10,6}$. The additional/reduction of time after remove/add process $= -d(v_{10}, v_8) + d(v_{10}, v_6) = -16 + 12 = -4$
vii. Remove $e_{23,6}$ and add $e_{23,8}$. The additional/reduction of time after remove/add process $= -d(v_{23}, v_6) + d(v_{23}, v_8) = -11 + 15 = 4$

Figure 2. The removed edges (in red colour) and the added edges (in green colour).

Thus, reduce time for the revised tour is $= -3 - 1 + 1 - 1 + 0 + -4 + 4 = -4$ so that the total time for the new tour is 248 – 4 = 244 minutes. Figure 3 shows the new tour obtained after deleting the crossing path.

Figure 3. The new tour after deleting the crossing path.

3.2 Solving with Modification of Sollin's Algorithm

The first step in Modification of Sollin's Algorithm is to connect each vertex in the graph with the smallest edge weight that incident to it. In this step, we get the following edges that connect the vertices in graph for each vertex. Table 3 shows the first step in the Modification of Sollin's Algorithm.

Vol. 6 No. 1, 2025, pp. 92-102

DOI: 10.23960/pepadun.v6i1.265

Table 3. The first step of Modification of Sollin's algorithm.

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
The smallest incidence edge	$e_{1,13}$	$e_{2,21}$	$e_{3,7}$	$e_{4,11}$	$e_{5,24}$	$e_{6,8}$	$e_{7,2}$	$e_{8,6}$	$e_{9,14}$	$e_{10,20}$	$e_{11,24}$	$e_{12,13}$
Vertex	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}
The smallest incidence edge	$e_{13,12}$	$e_{14,16}$	$e_{15,2}$	$e_{16,14}$	$e_{17,21}$	$e_{18,24}$	$e_{19,20}$	$e_{20,19}$	$e_{21,2}$	$e_{22,10}$	$e_{23,10}$	$e_{24,11}$

On the first step, we obtain six components as presented in Figure 4. Next, we search for the vertices that have degree more than two from the first step, which are v_2 , v_{10} and v_{24}. For v_2, we choose the highest edges that incidence to v_2 which is $e_{2,7}$ as the candidate edge to be remove (which implies that v_7 will become isolated vertex). Then, choose the vertex whose degree one that has smallest edge if we connect it to v_7, and that edge is $e_{7,15}$. Therefore, we remove $e_{2,7}$ and add $e_{7,15}$. We do the similar process with vertices v_{10}, and v_{24} so that we obtained the revised components as presented in Figure 5, with the list of edges to be selected to connect the leaves making all of the vertices two degree shown in Figure 6.

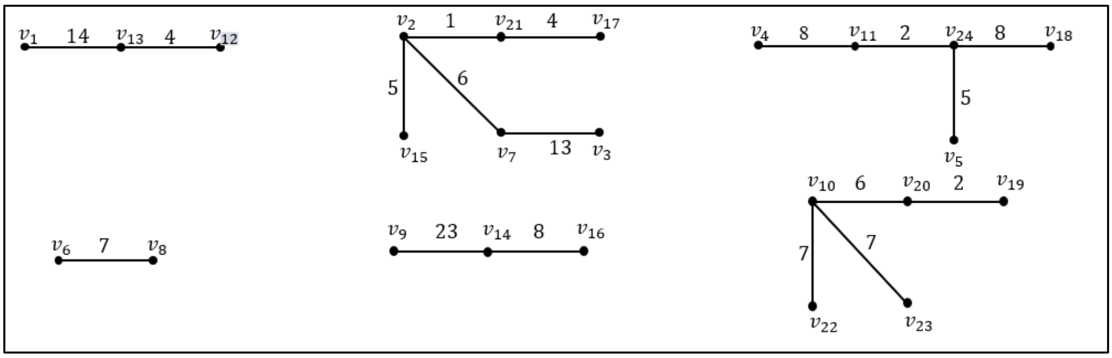


Figure 4. The six components obtained from the first step.

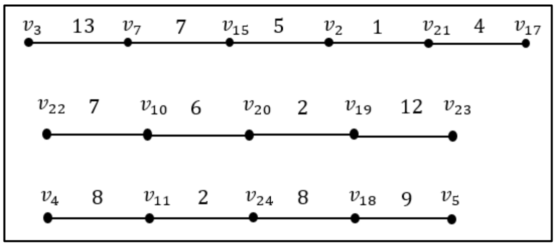


Figure 5. The revised component after the reducing degree of vertices with degree more than two.

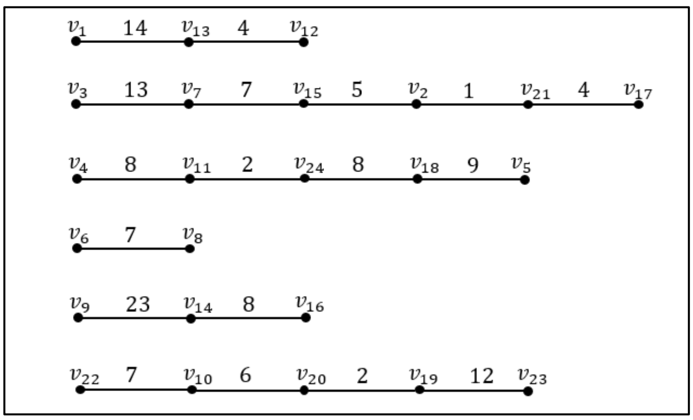


Figure 6. The list of edges to be selected to connect the leaves so that all leaves become vertices of degree two.

Vol. 6 No. 1, 2025, pp. 92-102

DOI: 10.23960/pepadun.v6i1.265

The next step is to connect the vertices whose degree one with the smallest possible edges chosen. Note that we are not doing anything with the vertices whose degree two. There are six components and, on every component, there are two vertices with degree one. Moreover, we are not allowed to connect the vertices whose degree one in the same component, because it will constitute a circuit. Table 4 shows the edges to be considered together with the weights.

Table 4. The edges to be considered together with the weights.

Edge	$e_{1,3}$	$e_{1,17}$	$e_{1,4}$	$e_{1,5}$	$e_{1,6}$	$e_{1,8}$	$e_{1,9}$	$e_{1,16}$	$e_{1,22}$	$e_{1,23}$	$e_{12,3}$	$e_{12,17}$	$e_{12,4}$	$e_{12,5}$	$e_{12,6}$	$e_{12,8}$
Weight	15	25	16	21	36	39	37	16	29	28	26	34	25	28	33	37
Edge	$e_{12,9}$	$e_{12,16}$	$e_{12,22}$	$e_{12,2}$	$e_{3,4}$	$e_{3,5}$	$e_{3,6}$	$e_{3,8}$	$e_{3,9}$	$e_{3,16}$	$e_{3,22}$	$e_{3,23}$	$e_{17,4}$	$e_{17,5}$	$e_{17,6}$	$e_{17,8}$
Weight	25	18	30	24	15	18	34	36	43	18	25	27	20	21	24	26
Edge	$e_{17,9}$	$e_{17,16}$	$e_{17,22}$	$e_{17,23}$	$e_{4,6}$	$e_{4,8}$	$e_{4,9}$	$e_{4,16}$	$e_{4,22}$	$e_{4,23}$	$e_{5,6}$	$e_{5,8}$	$e_{5,9}$	$e_{5,16}$	$e_{5,22}$	$e_{5,23}$
Weight	41	24	10	18	26	29	38	13	23	20	24	27	35	14	22	17
Edge	$e_{6,9}$	$e_{6,16}$	$e_{6,22}$	$e_{6,23}$	$e_{8,9}$	$e_{8,16}$	$e_{8,22}$	$e_{8,23}$	$e_{9,22}$	$e_{9,23}$	$e_{16,22}$	$e_{16,23}$				
Weight	33	22	17	11	37	26	21	15	36	25	21	14				

Choose the smallest edge $e_{17,22} = 10$, and then connect so that we get Figure 7(a). Continuing the similar process, we choose $e_{6,23} = 11, e_{1,3} = 15$ and obtain Figure 7(b). Then we choose $e_{4,16} = 13$ to obtain Figure 7(c).

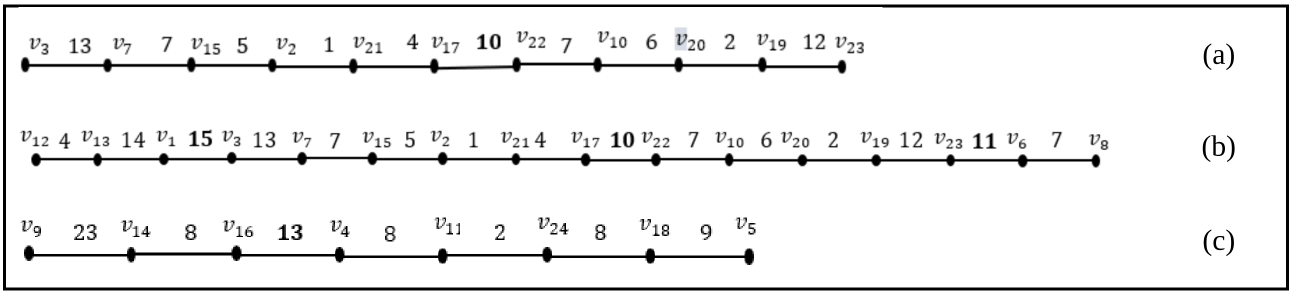


Figure 7. The components resulted in each step from Modification of Sollin's Algorithm.

Therefore, the six components are reduced into two components which are: the component consist of vertices v_3 and v_{23} as the leaves (vertices of degree one) and components that consist of vertices v_9 and v_5 as the leaves. The next step is searching the smallest edges that connect he leaves of each component, i.e selecting two of these four edges: $e_{5,8} = 27, e_{5,12} = 29,\ e_{8,9} = 37,\ e_{9,12} = 25$. By choosing $e_{9,12} = 25$ and $e_{5,8} = 2$, the solution is obtained, as shown in Figure 8.

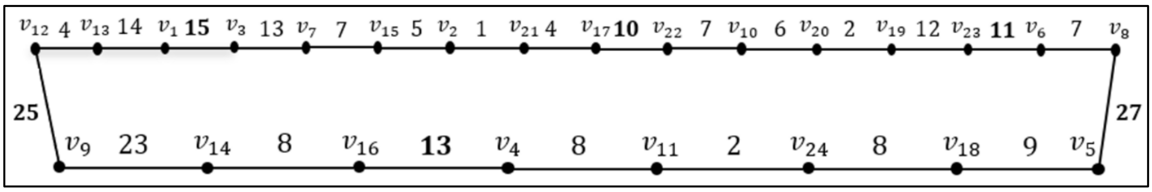


Figure 8. The final solution from Modification of Sollin's Algorithm.

From Figure 9 it can be seen that the solution consists of crossing intersection path, then the solution obtain are suboptimal and we refine the solution by removing intersection edges and adding other edges to obtain a new tour with better solution. Figure 10 shows the removed edges (in red colour), and the added edges (in green colour). There are some intersection paths in Figure 10, thus a deletion process is conducted, which resulted the final tour as presented in Figure 11.

DOI: 10.23960/pepadun.v6i1.265

Figure 9. The solution obtained manually using Modified Sollin's algorithm.

Figure 7. The removed edges (in red colour) and the added edges (in green colour).

Figure 8. The new tour after deleting the intersection path.

DOI: 10.23960/pepadun.v6i1.265

The addition and deletion of edges on the tour is done as follows:

i. Remove $e_{19,23}$ and add $e_{19,6}$. The additional/reduction of time after remove/add process $= -d(v_{19}, v_{23}) + d(v_{19}, v_6) = -12 + 18 = 6$
ii. Remove $e_{23,6}$ and add $e_{23,5}$. The additional/reduction of time after remove/add process $= -d(v_{23}, v_6) + d(v_{23}, v_5) = -11 + 17 = 6$
iii. Remove $e_{8,5}$ and add $e_{8,23}$. The additional/reduction of time after remove/add process $= -d(v_8, v_5) + d(v_8, v_{23}) = -27 + 15 = -12$

Thus, the time for the new tour is the same as original tour, because the process of removing and adding edges constitute zero time, which is $= -d(v_{19}, v_{23}) + d(v_{19}, v_6) - d(v_{23}, v_6) + d(v_{23}, v_5) - d(v_8, v_5) + d(v_8, v_{23}) = 6 + 6 - 12 = 0$.

4. CONCLUSIONS

Based on the above discussion the shortest tour obtained from both algorithms is 244 minute or 4 hours 4 minutes using the CIH Algorithm, while using the modified Sollin Algorithm the shortest path is 241 minutes or 4 hours 1 minute. From the results obtained, it can be concluded that the modified Sollin Algorithm is better than the CIH Algorithm for the case of solving TSP travel time for tourist attractions in Bandar Lampung City.

LITERATURE

[1] A. Gohil, M. Tayal, T. Sahu, & V. Sawalpurkar, "Travelling Salesman Problem: Parallel Implementations & Analysis", *arXiv preprint arXiv:2205.14352*, 2022.

[2] E. L. Lawler, "*The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization"*. John Wiley & Sons, 1985.

[3] M. Kurniawan, F. Farida, & S. Agustini, "Rute Terpendek Algoritma Particle Swarm Optimization dan Brute Force untuk Optimasi Travelling Salesman Problem", *Jurnal Teknik Informatika*, 14(2), pp. 191-200, 2021.

[4] S. Violina, "Analysis of Brute Force and Branch & Bound Algorithms to solve the Traveling Salesperson Problem (TSP)", *Turkish Journal of Computer and Mathematics Education*, 12(8), pp. 1226-1229, 2021.

[5] A. Wilson. , Y. Sibaroni, & I. Ummah, "*Analisis Penyelesaian Traveling Salesman Problem Dengan Metode Brute Force Menggunakan Graphic Processing Unit"*, 2(1), pp. 1874-1883, 2015.

[6] M. Amelia, I. I. Sholeha, Y. R. Revangga, & Wamiliana, "The Comparison of Brute Force, Cheapest-Insertion, and Nearest-Neighbor Heuristics for Determining the Shortest Tour for Visiting Malls in Bandar Lampung", *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, 14(1), pp. 51-54, 2024, doi: http://dx.doi.org/10.36448/expert.v14i1.3715.

[7] S. Hougardy & Wilde, "On the nearest neighbour rule for the metric traveling salesman problem", *Discrete Applied Mathematics*, 195, pp. 101-103, 2015.

[8] Winda A. F. B, S. R. Sumardi, N. N. Sari, & J. E. Simarmata, "Product Distribution Route using Nearest Neighbor Algorithm", *MALCOM: Indonesian Journal of Machine Learning and Computer Science*", 4(3), pp: 894-900, 2024.

[9] M. Y. Aswin, Wamiliana, Fitriani, M.Ansori, & Notiragayu, "Perbandingan Cheapest Insertion Heuristic dan Algoritma Christofides untuk Menentukan Tour Pasar Tradisional di Kota Bandar Lampung", *Jurnal Pepadun*, 5(2), pp. 182-194, 2024.

[10] L. V. Hignasari L & E.D. Mahira, "Optimization of Goods Distribution Route Assisted by Google Map with Cheapest Insertion Heuristic Algorithm (CIH)", *Jurnal Sinergi*, 22(2), pp. 132-38, 2018.

DOI: 10.23960/pepadun.v6i1.265

[11] K Meliantri, Githa, P. & N. K. A. Wirdiani, "Optimasi Distribusi Produk Menggunakan Metode Cheapest Insertion Heuristic Berbasis Web", *Jurnal Jurusan Teknologi Informasi*, 6(3), pp. 204-213, 2018.

[12] G. R. Utomo, S.D. Maylawati, & N. C. Alam, "Implementasi Algoritma Cheapest Insertion Heuristic (CIH) dalam Penyelesaian Travelling Salesman Problem (TSP)". *Jurnal Online Informatika*, 3(1), pp. 61- 67, 2018.

[13] Kusrini & J. E. Istiyanto, "Penyelesaian Travelling Salesman Problem Dengan Algoritma Cheapest Insertion Heuristics dan Basis Data", *Jurnal Informatika University Petra Kristian,* 8(2), pp. 109-114, 2007.

[14] F. J. Tjoea, & S. Halim, "Evaluasi Rute Pelayaran Kapal dengan Pendekatan Modified Minimum Spanning Tree", *Jurnal Titra*, 11(2), pp. 265-272, 2023.

[15] Wamiliana,"*Minimum Spanning Tree dan Desain Jaringan*", Pustaka Media, Bandar Lampung, 2022.